Journal of Intelligent Manufacturing

Hybrid genetic algorithms based on bin packing strategy for the unrelated parallel workgroup scheduling problem

--Manuscript Draft--

Manuscript Number:	JIMS-D-19-00463R4
Full Title:	Hybrid genetic algorithms based on bin packing strategy for the unrelated parallel workgroup scheduling problem
Article Type:	Original Research
Keywords:	Unrelated parallel machine problem; Workgroup scheduling; Two-dimensional bin packing problem; Heuristic strategy; Genetic Algorithm
Corresponding Author:	Naiming Xie, Ph.D Nanjing University of Aeronautics and Astronautics College of Economics and Management CHINA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Nanjing University of Aeronautics and Astronautics College of Economics and Management
Corresponding Author's Secondary Institution:	
First Author:	Bentao Su
First Author Secondary Information:	
Order of Authors:	Bentao Su
	Naiming Xie, Ph.D
	Yingjie Yang
Order of Authors Secondary Information:	
Funding Information:	National Natural Science Foundation of China (71671090) — Dr. Naiming Xie
	National Natural Science Foundation of China (71871117) — Dr. Naiming Xie
	Joint research project of National Natural Science Foundation of China and Royal Society of UK (71811530338) — Dr. Naiming Xie
	Fundamental Research Funds for the Central Universities (NP2018466) — Dr. Naiming Xie

Hybrid genetic algorithm based on bin packing strategy for the unrelated parallel workgroup scheduling problem

Bentao Su[a], Naiming Xie[a]* and Yingjie Yang[b]

a. College of Economics and Management, Nanjing University of Aeronautics and Astronautics, Nanjing, China

b. Institute of Artificial Intelligence, De Montfort University, Leicester, UK

*Correspondence: Naiming Xie, College of Economics and Management, Nanjing University of Aeronautics and Astronautics, 29th, Jiangjun Avenue, Nanjing, Jiangsu, People's Republic of China

Email: xienaiming@nuaa.edu.cn

Hybrid genetic algorithms based on bin packing strategy for the unrelated parallel workgroup scheduling problem

In this paper we focus on an unrelated parallel workgroup scheduling problem where each workgroup is composed of a number of personnel with similar work skills which has eligibility and human resource constraints. The most difference from the general unrelated parallel machine scheduling with resource constraints is that one workgroup can process multiple jobs at a time as long as the resources are available, which means that a feasible scheduling scheme is impossible to get if we consider the processing sequence of jobs only in time dimension. We construct this problem as an integer programming model with the objective of minimizing makespan. As it is incapable to get the optimal solution in the acceptable time for the presented model by exact algorithm, meta-heuristic is considered to design. A pure genetic algorithm based on special coding design is proposed firstly. Then a hybrid genetic algorithm based on bin packing strategy is further developed by the consideration of transforming the single workgroup scheduling to a strip-packing problem. Finally, the proposed algorithms, together with exact approach, are tested at different size of instances. Results demonstrate that the proposed hybrid genetic algorithm shows the effective performance.

Keywords: Unrelated parallel machine problem; Workgroup scheduling; Two-dimensional bin packing problem; Heuristic strategy; Genetic algorithm

1: Introduction

The effective management for modern manufacturing enterprises is essentially the reorganization and relocation of the existing resources of enterprises especially in the face of turbulent requirements. A certain proportion of human labor is prevalent in most enterprises due to the requirements on high quality and even the main mode of production in some specific industry processes is still carried out manually. Therefore, it is particularly important to optimize the allocation of human resources.

In a practical production system, the employees with single technical ability still account for a large proportion. In addition, many jobs involve more than one worker because of the complexity of their associated processes. These tasks require several personnel to work together as a workgroup, which is a group of personnel with complementary work skills for the tasks. Each workgroup has the ability to process one or more types of jobs independently. Therefore, assigning tasks often becomes a troublesome problem for managers not only to choose workgroup to process, but also to allocate workers under limited resource.

Consider such a scheduling problem if a set of jobs are processed by a number of unrelated parallel workgroups with fixed quantity of personnel in every group and each job has to choose one of the eligible groups to process with a determined processing time. Meanwhile, each job needs one or more personnel to process, which means it is possible that multiple jobs are processed simultaneously in one workgroup. However, these jobs can’t be processed infinitely at the same time because of the limited total number of personnel in each workgroup. This scheduling problem is common especially in complex equipment manufacturing areas, such as aerospace composites manufacturing and locomotive production, etc.

The classical unrelated parallel machine scheduling problem with additional resource (UPMR) refers to a set of jobs arranged to a number of parallel machines which require a number of units of a scare resource and there's a total amount of resources. This problem requires that no more than the total number of resources are used at any time. As for the problem we deal with in this paper, it’s similar to the UPMR problem when we regard workgroup as machine and the number of personnel for each workgroup is the human resource. We refer to such scheduling problem as the unrelated parallel workgroup scheduling problem with eligibility workgroup and resource constraints (denoted by UPWR).

The rest of the paper is organized as follows. In section 2, we review the relevant research work. In section 3, we define the problem and present the mathematical formulations. In section 4, we propose the two meta-heuristic algorithms to solve this problem. In section 5, the proposed algorithms are computationally tested and compared with existing models. Finally, we conclude the study and discuss for future research in section 6.

2: Literature Review

The classical parallel machine scheduling problem (PMSP) is a typical scheduling problem in textile industry, electronic manufacturing, mechanical processing, etc (Pinedo 2016). In 1959, PMSPs were firstly proposed by McNaughton (1959) and have attracted wide attention from scholars since then. Mathematically, even the two identical parallel machines were also demonstrated to be NP-hard (Lenstra, Rinnooy Kan, and Brucker 1977). According to the different types of parallel machines, PMSPs can be generally classified into identical (Lann and Mosheiov 2003), uniform (Lee, Chuang, and Yeh 2012) and unrelated (Mokotoff and Chrétienne 2002). As the identical and uniform parallel machines can be regarded as special cases of unrelated parallel machines, the UPM problem is a general form of PMSP.

In most of the UPM study, machine is the only resource to be considered. However, in a real-world production system, processing a job may needs another resource such as human labor, materlals, energy and so on (Slowinski 1980; Blazewicz et al. 1983; Ventura and Kim 2000). After that, the research on UPMR problem mainly focuses on the following aspects: types of resources, objective functions and solution methods. The types of resources are renewable, non-renewable and doubly constrained (Edis, E.B., Oguz, C., and Ozkarahan, I. 2013). Chen, L., Ye, D., Zhang G. (2018) consider a problem of scheduling with renewable speed-up resources and give a 2-

approximation algorithm. Due to the consideration of additional resources, problems based on different objective functions have also been studied. Edis, E.B., Oguz, C., and Ozkarahan, I. (2012) proposed integer and constraint programming models for minimizing the completion time of the last job.

Solution approaches related to the UPMR problems are mainly included polynomial-time algorithms, exact approaches and approximation/heuristic approaches. Gyorgyi, Peter (2017) propose a polynomial time approximation scheme to solve the UPMR with non-renewable resource constraints. Fu, Y., Jiang, G., Tian, G., et al (2019) design a constructive heuristic approach and a hybrid nested partition method to fix the UPS where the resource needs to be allocated to machines in advance. Fanjul-Peyro, L., Perea, F., Ruiz, Rubén (2017) uses the ideas obtained from bin packing problem and propose matheuristic strategies. In addition, Akyol Ozer, E., Sarac, T. (2018) and Afzalirad, M. and Shafipour, M. (2018) also use meta-heuristic algorithm to solve the problem for large size of instances. More recently a lot of researches have begun to focus on the changes of time related to resources. Jin, J., Ji, P. (2017) consider that the resource-dependent ready times of jobs are continuous functions of their consumed resource. Wang, Z., Xiao, C., Lin, X., and Lu, Y. (2017) consider a single-machine scheduling problem with a deteriorating and resource-dependent maintenance activity.

At present, most of the researches on personnel scheduling are transformed into machine scheduling problems. Considering the particularity of UPWR problem, newer methods need to be considered on the basis of traditional approaches.

3: Problem definition and mathematical formulations

This study describes an unrelated parallel workgroup scheduling problem with eligibility and resource constraints (UPWR), where there are a set of jobs to be allocated to a set of workgroups with determined processing time and number of processing personnel. The workgroups are unrelated due to the different work skills of the staff. The difference in work skills results in the inability of personnel to move between workgroups, as well as the impossible processing once the manpower is insufficient. The goal is minimizing the makespan.

3.1: Assumptions

- All jobs are available at time zero, the processing time and number of personnel for each job, are fixed and known in advance.
- All workgroups are always available for processing since time zero.
- Each workgroup can process one or more jobs at a time within resource constraints.
- Workgroup eligibility constraints: not all jobs can be processed on all workgroups.
- Preemption is not allowed.

3.2: Notations

For convenience, following notations are introduced.

(1) Indices

$i = 1, 2, ..., m$	Index for workgroups
$j = 1, 2, ..., n$	Index for jobs
$t = 0, 1, ..., T_{\max}$	Index for time

(2) Problem Parameters

p_{ij}	Processing time of job j on workgroup i
eg_{ij}	1 if workgroup i capable to process job j ; 0, otherwise
r_{ij}	Number of processing personnel of job j on workgroup i
R_i	Total number of human resources on workgroup i

(3) Decision Variables

x_{ijt}	1 if job j is assigned to workgroup i and completes its processing at time t ; 0, otherwise

$C_{\max}$ makespan

3.3: Mathematical formulation

There, the UPWR is considered as a mixed integer programming model. It is formulated as below:

(Problem-UPWR)

$$Min\ C_{\max} \tag{1}$$

subject to

$$\sum_{i=1}^{m}\sum_{t=p_{ij}}^{T_{\max}} x_{ijt}\cdot t \le C_{\max} \quad \forall\ \mathrm{j} \tag{2}$$

$$\sum_{i=1}^{m}\sum_{t=p_{ij}}^{T_{\max}} x_{ijt} = 1 \quad \forall\ \mathrm{j} \tag{3}$$

$$\sum_{t=p_{ij}}^{T_{\max}} x_{ijt} \le eg_{ij} \quad \forall\ \mathrm{i,j} \tag{4}$$

$$\sum_{j=1}^{n}\ \sum_{s=\max\{t,p_{ij}\}}^{\min\{t+p_{ij}-1,T\}} r_{ij}\cdot x_{ijs} \le R_i \quad \forall\ \mathrm{i,t} \tag{5}$$

$$x_{ijt} \in \{0,1\} \quad \forall\ i,j,t \tag{6}$$

There is one objective function of the UPWR, which aims to minimize the makespan (Eq. 1). Formulas (2) to (6) are constraints. Constraints (2) determine the makespan. Constraints (3) dictate that each job is assigned exactly one workgroup. Constraints (4) are workgroup eligibility constraints where a job is not allowed to be processed on an ineligible workgroup. Constraints (5) ensure that no more than R_i units of human resource are used at any time for each workgroup. Constraints (6) indicates that the decision variables x_{ijt} are binary.

In order to give a clearer illustration of the UPWR, an example is given to show the differences between this problem and the UPMR.

Example 1.1. Consider the following instance of eight tasks ($n=8$) needed to be processed. There are two available machines $M1,\ M2$ for the UPMR with total eight units resources ($R_{\max}=8$) , while there are two available workgroups $W1,\ W2$ with four units of personnel ($R_1=R_2=4$)in each group for the UPWR. Let the processing time is denoted by p_{ij} and the number of resource need is r_{ij} .The specific processing data are shown in Table 1. Some columns have null values in table 1, which means that the corresponding task can't be processed by this machine or workgroup.

Table 1. Processing data of example 1.1

Job j	p_{1j}	p_{2j}	r_{1j}	r_{2j}
1	3	NULL	2	NULL
2	NULL	3	NULL	3
3	3	2	2	2
4	3	3	4	1
5	3	2	3	4
6	1	2	1	1
7	NULL	1	NULL	1
8	1	2	4	3

Figure 1 shows the solutions to UPMR and UPWR. An optimal solution of the UPMR is shown in Fig.1a, where each machine is fully utilized and the optimal makespan $C_{\max}=8$, but we can see the resources are not fully utilized. If these jobs are processed by workgroups, we obtain the solution given in Fig.1b with a shorter optimal makespan $C_{\max}=5$ and the utilization rates of personnel in the two workgroups are 95% and 100% respectively. As we can see, in UPWR problem, different jobs can be processed in one workgroup at the same time, for example, job 3 and 1, job 2 and 4. By processing multiple tasks at the same time in one workgroup, the completion time is greatly shortened and the utilization of resources is effectively improved.

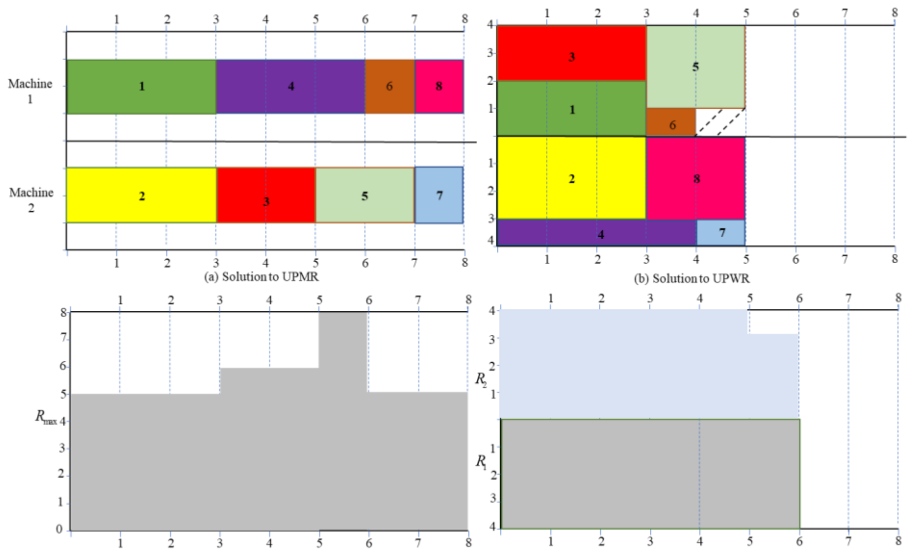

Figure 1. Gantt diagrams representing the solutions to UPMR and UPWR

Through the analysis of example1.1, the UPWR problem needs to consider not only the assignment of jobs to workgroups, but also the allocation of resources of each workgroup to jobs. Therefore, it requires another way of thinking so as to solve this problem.

4: Solution method

Solving the UPWR problem consists in determining the assignment of the jobs to the workgroups under workgroup eligibility constraints and the processing position of the jobs in the selected workgroup based on the constraints of human resources. In particular, it is possible for a workgroup to process multiple jobs at a time, which means that it is needed to confirm the processing sequence of jobs to a workgroup searching in a two-dimensional solution space. Different from the classical UPMR problem where the determination of order of jobs to a machine is only considered in time dimension, the UPWR is more complicated and difficult to solve.

As the addressed problem is NP-hard, applying exact approaches to solve the UPMR instances, especially for large-scale problems, may not get satisfactory solution within an acceptable time. Hence, we focus on developing meta-heuristic algorithms to solve it. Genetic algorithm (GA), proposed by Holland and John, H. (1973), is a well-known approach to generate near-optimal solutions with flexible encoding scheme and genetic operators in a short computation time, which are widely used in various optimization problems (Rubén Ruiz, Concepción Maroto 2006, Ta, Q.C., Billaut, J.C., Bouquard, J.L. 2015). Besides, GA has been applied to solve the parallel machine scheduling problems.

In this section, we first define a kind of mapping to map a set of processing sequences of jobs to a workgroup into a feasible scheduling scheme and propose a pure genetic algorithm with chromosome encoding of two-dimensional vector based on this mapping. In addition, considering the large solution space of the job processing sequence, we use the ideas obtained from bin-packing strategy to optimize the population initialization, crossover operator and mutation operator.

4.1: Genetic algorithm (GA)

4.1.1: Encoding and decoding operator

For the UPMR problem, a set of processing sequences of jobs to a machine can represent a scheduling scheme on the corresponding machine. It is not feasible to represent a scheduling scheme directly by using the order of jobs as multiple jobs can be processed in a workgroup at one time. Hence, a two-dimensional vectors group coding method is adapted to construct chromosome and a mapping rule is designed to ensure the uniqueness of decoding operator.

Let $Ch = [g_1, g_2, ..., g_n] = \begin{bmatrix} i_1, i_2, ..., i_n \\ s_{i_1}, s_{i_2}, ..., s_{i_n} \end{bmatrix}$ denotes a chromosome where the length of it is equal to the total number of jobs n. For each gene $g_j = \begin{bmatrix} i_j \\ s_{i_j} \end{bmatrix}$ $(j = 1, 2, ..., n)$, i_j represents the selected workgroup of job j and s_{i_j} denotes the processing sequence of job j in workgroup i_j.

Without loss of generality, assume subset J_i consists of all the jobs assigned to the workgroup i. Let $\phi(J_i)$ denote a feasible scheduling scheme of workgroup i under a given processing sequence (denoted by $S(J_i)$) of job set J_i, where ϕ represents a mapping relationship. Let set E denote a set of time index t_k $(t_k = 0, 1, ..., T_{\max})$. Such a mapping process is achieved by the following rule (donated by MAP-Rule) based on the MIP model.

MAP-Rule

Input: Processing sequence $S(J_i)$ of job set J_i in workgroup i

Output: Scheduling scheme $\phi(J_i)$

```
Initializes the set $E=\{0\}$, decision variable $x_{ijt}=0$;
for $J \in J_i$ do
    for $t$ in $E$ do
        Let $x_{iJ(t+p_{iJ})} = 1$
        if $\forall$ k $\in \{t, t+p_{iJ}-1\}, \sum_{j=1}^{n} \sum_{s=\max\{k,p_{ij}\}}^{k+p_{ij}-1} r_{ij} \cdot x_{ijs} \le R_i$ , then
            add $t+p_{iJ}$ into set $E$
            sort $E$ with ascending order
        else
            Let $x_{iJ(t+p_{iJ})} = 0$
            delete $t$ from $E$
            continue
```

We use a diagram to depict the process of encoding and decoding operator by taking the data of Example 1.1, which is shown in figure 2. The first string includes the assigned workgroup which can process the corresponding job while the second is the processing sequence of each job in the corresponding workgroup. After adding other parameter data, we can decode this chromosome into the scheduling scheme based on MAP-Rule, which is shown in Fig.2.

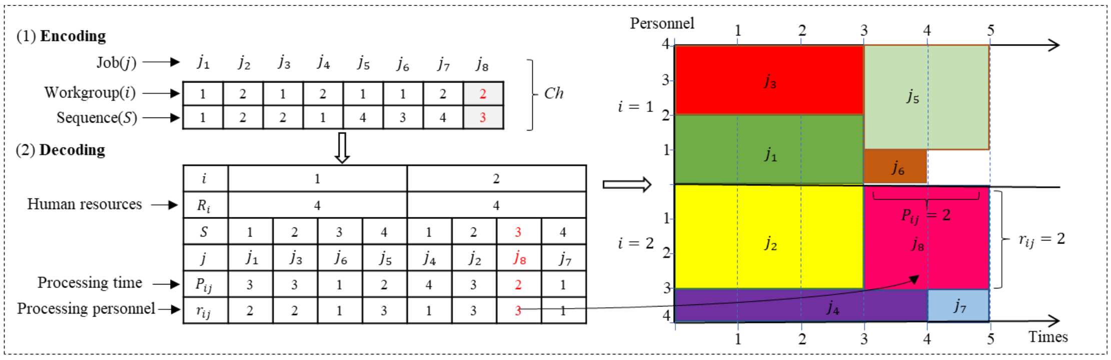

Figure 2. The diagram of encoding and decoding operator

4.1.2: Population initialization

The initial population is generated in a completely random way, the procedure is as follows:

Step1: For each job j, a_j is chosen randomly in set V_j, thus we obtain a n-dimensional vector $\{a_1, a_2, ..., a_n\}$ as the first string of chromosome;

Step2: For each workgroup i, count the total number of jobs N_i;

Step3: For each job j, the order of job j in workgroup i is a natural number which is selected randomly and without repetition from $[1, N_i]$. Then, we get the second string.

4.1.3: Crossover operator

In view of the special nature of the chromosome encoding and the fact that each gene string represents a different meaning, we need a special design on the crossover operation. Here the two rows of chromosomes are crossed respectively, which is shown in figure 3. And the steps of crossover operator are given as follow.

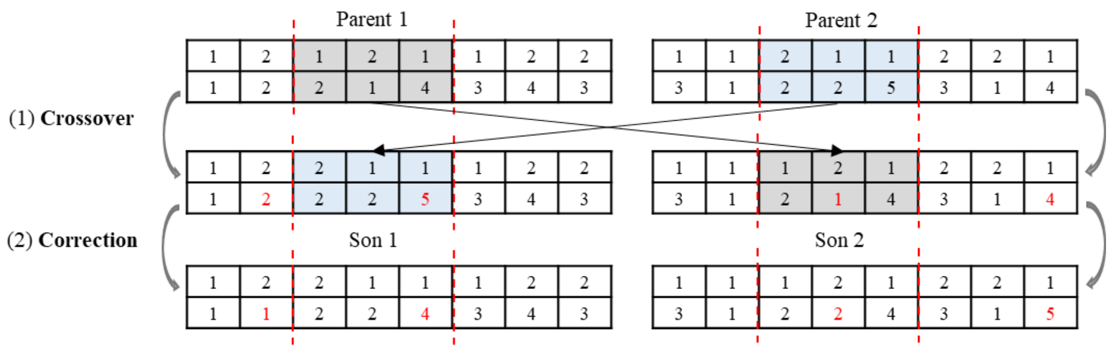

Figure 3. The diagram of crossover operator

Step1: Randomly select the same location gene fragments of two parent individuals and cross the first-row gene information;

Since the set of workgroups that each job can choose is predefined, the processing constraints are still satisfied after crossover. In order to make the next generation retain the parent information as much as possible, which is the processing sequence on each workgroup, the second line of gene string need to be modified with specific rules.

Step2: For each workgroup i, make the minimum value in the list of processing sequence for the corresponding workgroup in the second line of the gene string as 1, the second minimum value as 2, and so on. If there are p identical values corresponding to the new gene value q, then randomly select p values without repetition from $[q, q+p-1]$ to be the p gene values of the corresponding position after correction.

4.1.4: Mutation operator

Firstly, the gene in the first line of gene string is mutated in its optional workgroup set. Then, the second line of gene string is modified by cross-operation modification scheme. The diagram of mutation operator is shown in figure 4.

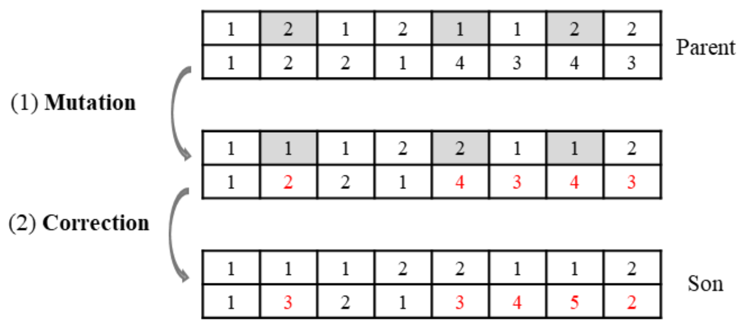

Figure 4. The diagram of mutation operator

4.2: Hybrid Genetic algorithm based on bin packing strategy

As for the pure genetic algorithm, it may lead to slow convergence speed and local optimum easily considering that the generation of processing sequence of jobs to a workgroup is completely random. Hence, a bin-packing strategy is used to generate the processing sequence of jobs for each workgroup, which can optimize the operations of population initialization.

4.2.1: Bin packing strategy

Consider a single workgroup scheduling problem where the jobs assigned to the workgroup are known. It can be regard as a two-dimensional strip packing problem, which is shown in figure 5. Given a rectangular case where the length represents the makespan and the width represents the units of human resources, the objective is to place a set of rectangular items (represent jobs) into the case with no overlap in x-axis or y-axis with the previously item j_k so that the length is minimized. Based on this, the single workgroup scheduling problem can be solved with packing methods. At present, a number of studies have used the idea of packing methods for solving scheduling problems (Liang, X., Zhou, S., Chen, H., and Xu, R. 2019; Fanjul-Peyro, L., Perea, F., and Ruiz Rubén 2017).

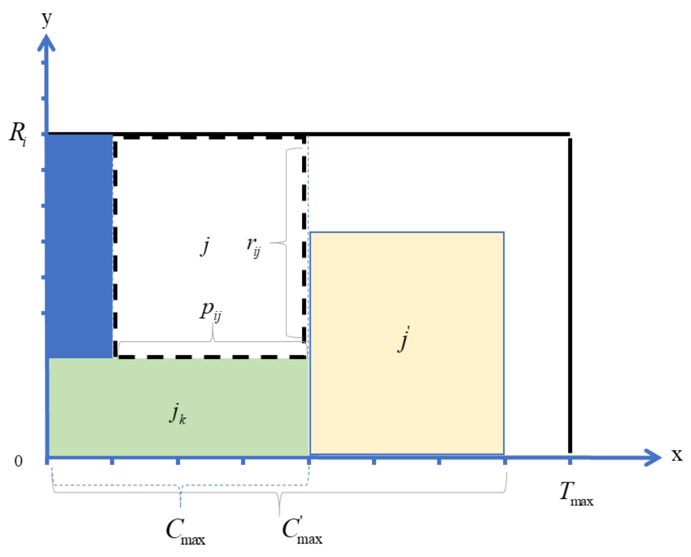

Figure 5. The diagram of strip packing problem

In this section, by analysing the best-fit heuristic algorithm proposed by Burke, E., Kendall, R., and Whitwell, G. (2006) and the recursive heuristic algorithm given by Peng, B.T., and Zhou, Y.W. (2012), we give a best priority first-heuristic packing strategy (denoted by BP) and apply it to solve single workgroup scheduling problem based on the proposed MIP model. There are seven kinds of priorities of job (denoted by P_j) which can be seen in figure 6 and the idea of this method is to find the highest priority job to assign in the process of each iteration.

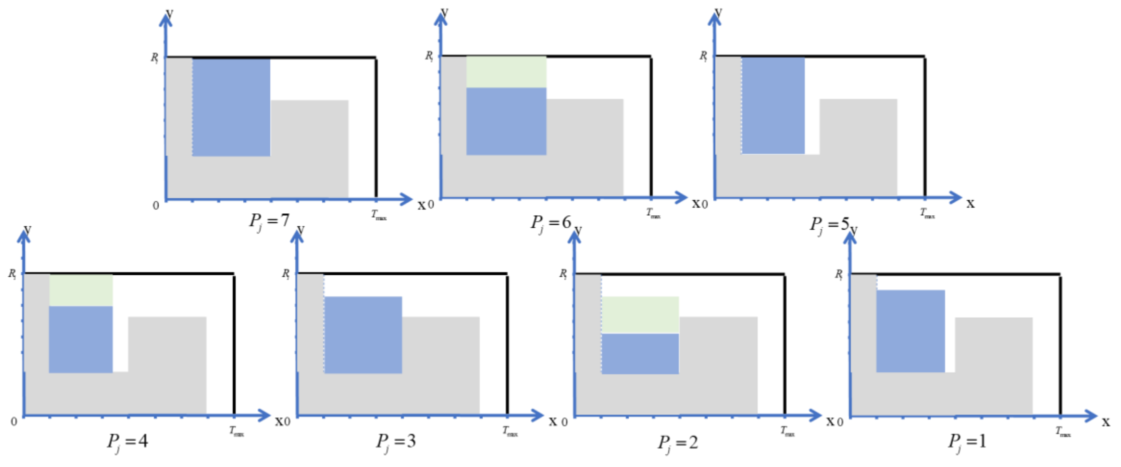

Figure 6. The rules of priority setting

Based on the above rules, the steps of BP algorithm are shown as follows.

Algorithm: BP

```
**Input**: Job set $J_i$ processed in workgroup $i$
**Output**: Processing sequence $S(J_i)$ of jobs in workgroup $i$.
  Initializes the node set $E=\{0\}$, decision variable $x_{ijt}=0$;
  Sort $J_i$ in descending order by $r_{ij} \cdot p_{ij}$;
  for $t_k \in E$ do
    if $\exists j \in J_i$, when $x_{iJ(t_k+p_{iJ})}=1$, $\sum_{j\in J_i}\sum_{s=\max\{t_k,p_{ij}\}}^{t_k+p_{ij}-1} r_{ij} \cdot x_{ijs} < R_i$, then
      Flag = True
    else
      Flag = False
    if Flag == False, then
      delete $t_k$ from $E$
      continue
    else if $\exists j \in J_i, r_{ij} = R_i - \sum_{j\in J_i}\sum_{s=\max\{t_k,p_{ij}\}}^{t_k+p_{ij}-1} r_{ij} \cdot x_{ijs}$, $p_{ij}=t_k-t_{k-1}$, then $P_j=7$
    else if $\exists j_1, j_2 \in J_i$, $r_{ij_1}+r_{ij_2} = R_i - \sum_{j\in J_i}\sum_{s=\max\{t_k,p_{ij}\}}^{t_k+p_{ij}-1} r_{ij} \cdot x_{ijs}$, $p_{ij_1}+p_{ij_2}=t_k-t_{k-1}$, then $P_{j_1,j_2}=6$
    else if $\exists j \in J_i$, $r_{ij} = R_i - \sum_{j\in J_i}\sum_{s=\max\{t_k,p_{ij}\}}^{t_k+p_{ij}-1} r_{ij} \cdot x_{ijs}$, then $P_j=5$
    else if $\exists j_1, j_2 \in J_i$, $r_{ij_1}+r_{ij_2} = R_i - \sum_{j\in J_i}\sum_{s=\max\{t_k,p_{ij}\}}^{t_k+p_{ij}-1} r_{ij} \cdot x_{ijs}$, then $P_{j_1,j_2}=4$
    else if $\exists j \in J_i$, $p_{ij}=t_k-t_{k-1}$, then $P_j=3$
    else if $\exists j_1, j_2 \in J_i$, $p_{ij_1}+p_{ij_2}=t_k-t_{k-1}$, then $P_{j_1,j_2}=2$
    else $P_j=1$
    Choose $j^* \in J_i$ and $P_{j^*}=\max\{P_j\}$ or choose $j_1^*, j_2^* \in J_i$ and $P_{j_1^*,j_2^*}=\max\{P_j\}$
    Let $x_{ij^*(t_k+p_{iJ})}=1$ and delete $j^*$ or $j_1^*, j_2^*$ from $J_i$
    update $E$
    if $J_i=\varnothing$, then
      break
    else
      continue
  end
```

4.2.2: Hybrid genetic algorithm

For a given working group, as long as the jobs processed in it are determined, it is obvious that the scheduling scheme generated by the processing sequence obtained by the BP algorithm is better than the randomly generated result in most cases. Based on the BP algorithm, a hybrid genetic algorithm (denoted by BP_GA) is proposed. In BP_GA, instead of generating the processing order of jobs randomly, the BP algorithm is added to produce better chromosomes. Except for the formation of initial population, the other process including the mutation and the crossover operations for GA are also adopted for BP_GA. The steps of BP_GA are given as follows.

Algorithm: BP_GA

```
Input: Data of jobs and workgroups
Output: A near optimal scheduling scheme $J$
  1. Parameter initialization: population size $S$, maximum generation $G_{\max}$, crossover
probability $P_c$, and mutation probability $P_m$
  2. Let generation index $k = 1$
  3. Population initialization:
     Let individual index $r = 1$
     while $r \le S$, do
        Determine $\{i_1, i_2, ..., i_n\}$, $i_j \in \{1, 2, ..., m\}$
        for each workgroup $i$:
           Get job set $J_i$
           Run BP with one input $J_i$
           return $S(J_i)$
        Generate chromosome $Ch_r$
      $r = r + 1$
     $\text{pop}(k=1) = \{Ch_1, ...Ch_r, ...Ch_s\}$
  4. while $k \le G_{\max}$, do
     Do decoding operations
     Calculate the fitness of the $k$ population
     Roulette selection
     if random<=$P_c$, then
        Do crossover operations for the first-row gene string
        Do correction operations for the second-row gene string
     if random<=$P_m$, then
        Do mutations operations for the first-row gene string
        Do correction operations for the second-row gene string
     $k = k + 1$
  5. Choose the population with the lowest fitness
  end
```

5: Computational experiments

In this section, we conduct computational experiments with instances of small and large sizes to evaluate the performance of the proposed GAs. For small problems, a branch and bound approach (B&B) under the Gurobi 8.1.1 software is adopted to get the optimal solution approach and compared with the proposed algorithms. However, with the increase of the size of the problem, it is impossible to get an optimal solution in an acceptable CPU time. Hence, the problems are optimally solved by the B&B with 3h run time limitation. All instances are randomly generated and the results are discussed for different sizes. The generation of instances and the proposed algorithms are coded with Python 3.7 by the PyCharm 2019 software. Both Gurobi and Pycharm software run on a personal computer including Inter(R) Core(TM) i5-8265U CPU with 2GHz speed and 8GB of RAM.

5.1: Instances generation

It is important that different degrees of parameters will affect the performance of the solution obtained by the algorithm. For the instances of UPWR problem, we choose the combination of the total number of workgroups (m) and the number of jobs (n) with different levels to reflect the size of the experiment. The other parameters for scheduling problems are purely at random within a given range. $U(a,b)$ is a random integer uniformly distribution between a and b (both extremes included), which is the most commonly distribution used for generating the instance about scheduling problem.

(1) The number of workgroups m : $m \in \{2,4,6\}$ is considered.

(2) The number of jobs n : $n \in \{10,20,30,40,50\}$ is considered for small-size instance and $n \in \{120,140,160,180,200\}$ for large-size.

(3) The maximum value of the time index $T_{\max}$: For the UPWR problem, $T_{\max}$ is an upper bound of the optimal solution. For a certain workgroup, the maximum complete time equal to the sum of the processing times of these jobs which can be processed in this workgroup. Hence, $T_{\max}$ can be set equal to the maximum value of the maximum complete times for all workgroups, that is,

$$T_{\max} = \max\left\{\sum_{j=1}^{n} eg_{ij} \cdot p_{ij}\right\}, i = 1, 2, ..., m \text{ .}$$

(4) The processing time p_{ij} of job j on workgroup i : let $p_{ij}=U(1,20)$.

(5) The eligible constraints eg_{ij} : for job j , first determine the number of workgroups which can process job j is $U(1,m)$, then randomly add a workgroup without repetition from set of workgroups $\{1,2,...,m\}$ to a set V_j and execute $U(1,m)$ times. Let $eg_{ij} = 1$ if $i \in V_j$ and $eg_{ij} = 0$ if $i \notin V_j$.

(6) The total number R_i of human resources on workgroup i : let $R_i=U(10,15)$.

(7) The number r_{ij} of processing personnel of job j on workgroup i : let $r_{ij}=U(1,R_i)$.

Each size has totally 9 test problems respectively (denoted by $n \times m$). For small-size instance, let $n \in \{10,20,30\}$ when $m=2$, $n \in \{20,30,40\}$ when $m=4$, and $n \in \{30,40,50\}$ when $m=6$. The setting for large-size instance is the same. In addition, we repeat all possible combinations ten times. Hence, the number of small-size and large-size instances to be tested is 90 separately.

5.2: Algorithm parameter setting

The performance of GA is generally sensitive to the settings of the parameters, which mainly include maximum generation $G_{\max}$, population size S , crossover probability P_c , and mutation probability P_m . In this section, design of experiment (DOE) (Quenouille, M.H., Kimball, A.W. 1953) is adopted to test the influence of different levels of factors on algorithm performance.

Each parameter takes four levels and the value of each level is shown in Table 2. According to the number of parameters and levels, an orthogonal experiment with $L_{16}\left(4^4\right)$ size is adopted in this paper. The $n \times m = 50 \times 4$ size of instance and pure genetic algorithm (GA) are selected to test, and the algorithm runs independently 20 times for each combination of parameters. The average value of relative percentage deviation (RPD) is applied for response variable (RV). The orthogonal table and RV value of each parameter combination are shown in Table 3, the range and importance of each parameter are shown in Table 4, and the influence trend of each parameter on algorithm performance is shown in Figure 7 where the gray line shows mean value of each parameter.

Table 2. Values of each level for four parameters

Parameter	Level			
	1	2	3	4
$G_{\max}$	100	150	200	300
S	100	150	200	250
P_c	0.6	0.7	0.8	0.9
P_m	0.01	0.05	0.1	0.25

Table 3. Values of each level for four parameters

No.	Level				RV
	$G_{\max}$	S	P_c	P_m	
1	1	1	1	1	42.84
2	1	2	2	2	33.86
3	1	3	3	3	29.09
4	1	4	4	4	24.43
5	2	1	2	3	36.47
6	2	2	1	4	30.22
7	2	3	4	1	27.61
8	2	4	3	2	24.55
9	3	1	3	4	30.34
10	3	2	4	3	25.57
11	3	3	1	2	29.54
12	3	4	2	1	31.36
13	4	1	4	2	31.59

14	4	2	3	1	32.95
15	4	3	2	4	25.34
16	4	4	1	3	26.25

Table 4. Values of each level for four parameters

Level	$G_{\max}$	S	P_c	P_m
1	32.56	35.31	32.22	33.69
2	29.72	30.65	31.76	29.89
3	29.20	27.90	29.23	29.35
4	29.03	26.65	27.30	27.59
Range	3.52	8.66	4.91	6.11
Rank	4	1	3	2

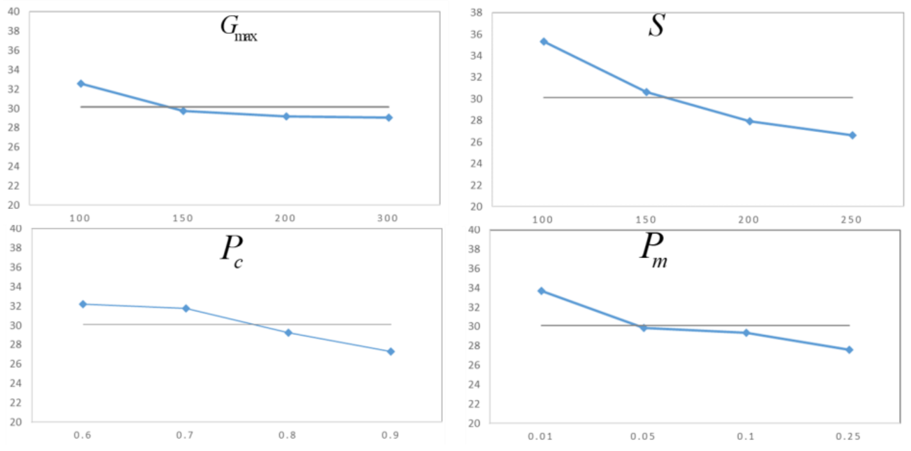

Figure 7. The influence trend of each parameter on algorithm performance

As shown in Table 3 and Fig. 7, the maximum range is population size S, which indicates different population sizes have great influence on the algorithm. Small value of S will get poor result while large value will affect the search efficiency. The second largest is mutation probability P_m. As with population size, the value of is P_m neither too large nor too small. The maximum generation $G_{\max}$ and crossover probability P_c hold small fluctuation during the test. Based on the above analysis, the recommended values of the four parameters are: $G_{\max}$=150, S=200, P_c=0.8, P_m=0.1.

5.3: Experimental results and analysis

To evaluate the performance of the proposed algorithms, the CPU time and RPD are often used as reference indexes. Since all size of problems are tested for ten times, we use the average CPU time (ACT) and the relative percentage deviation ($ARPD$) to evaluate each algorithm. $ARPD$ is computed for each instance according to the following expression:

$$ARPD = 100 \times \left(C_{\max}^{a\lg} / C_{\max}^{\min} - 1 \right) / 10 \tag{7}$$

where $C_{\max}^{a\lg}$ is the makespan obtained by each algorithm for each size of instance and $C_{\max}^{\min}$ is the best solution through all approaches.

The test results for the UPWR problem with small sizes and algorithms are summarized in Table 5. The $ARPD$ and ACT by ten times running of B&B, GA, BP_GA for small-size instance are compared.

Table 5. Experimental results given by the three algorithms for small-size problem

No.	Size	ARPD			ACT (s)		
	$n \times m$	B&B	GA	BP_GA	B&B	GA	BP_GA
1	10×2	0	0	0	1.01	5.5	5.9
2	20×2	0	0	0	6.98	11.5	17.4
3	30×2	0	9.13	7.97	508.28	15.5	16.2
4	20×4	0	8.42	6.32	0.72	19.9	24.4
5	30×4	0	19.6	25.2	43	16.9	16.3
6	40×4	0	18.85	20.86	2025.42	16.9	42.2
7	30×6	0	19.05	20.48	2.04	17.6	12.8
8	40×6	0	12.58	1.61	3.35	17	22.4
9	50×6	0	31.2	42.8	10.32	25	32

Table 6. Comparison between GA and BP_GA for large-size problems

No.	Size	ARPD		ACT (s)	
	$n \times m$	GA	BP_GA	GA	BP_GA
1	120×2	4.43	3.58	75.5	58.5
2	140×2	7.59	4.99	89.3	83.5
3	160×2	8.95	3.62	99.6	99.3
4	140×4	12.76	18.09	80.7	90.5
5	160×4	8.56	5.10	116	92.7

6	180×4	6.33	4.25	128.4	98.9
7	160×6	16.72	10.00	110.8	93.5
8	180×6	15.07	11.58	121.7	117.3
9	200×6	16.16	9.80	126.5	108.8

Table 5 shows the obtained results from B&B, GA and BP_GA for nine size problems. As it can be seen from Table 5, the proposed Gas can get satisfactory solution in a short time but the exact algorithm can also get the exact solution in a very short time for most cases. Hence, the meta-heuristic algorithm has no advantage over exact algorithm in solving small-scale problems. However, for large size of problems, as shown in Table6, the exact algorithm can’t achieve best solution in acceptable run time. But the GAs can get a better solution in a short time and BP_GA performs better than GA. As a result, for selecting only one algorithm, BP_GA is more suitable than GA to finding the accurate solution.

6: Conclusions

This paper addresses a realistic unrelated parallel workgroup scheduling problem where tasks are processed by workgroups rather than machines which results in the ability to process multiple tasks in the same workgroup at the same time. An integer programming model is proposed to consider makespan for this problem. A pure genetic algorithm and a hybrid genetic algorithm based on bin packing strategy are given, and different sizes of instances are adopted to test proposed algorithms. The main contributions could be summarized as follows:

(1) An unrelated parallel workgroup scheduling problem with workgroup eligibility and resource constraints is proposed where a workgroup process multiple jobs at the same time, and we formulate the problem as an integer programming model with the objective of minimizing makespan.

(2) A pure genetic algorithm (GA) model is adopted to solve this problem. In order to describe this problem easily, we design a mapping rule to simplify chromosome coding.

(3) A hybrid genetic algorithm based on bin packing strategy is further developed by the consideration of transforming the single workgroup scheduling to a strip-packing problem, i.e. total amount of personnel in a workgroup is defined as width of rectangle and makespan represents the length of rectangle.

(4) Small and large sizes of cases are adopted to test the algorithms. Results show that the proposed BP_GA can provide a feasible and superior solution for the UPWR

Additionally, our future research directions involve the consideration of other factors such as the release times and due dates of jobs. Furthermore, we will build multi-objective optimization model in future studies in order to make our model more realistic. Considering other bin packing strategies also seems interesting.

Acknowledgements:
This work was supported by National Natural Science Foundation of China under grant 71671090 and 71871117, Joint research project of National Natural Science Foundation of China and Royal Society of UK under grant of 71811530338, the Fundamental Research Funds for the Central Universities under grant NP2018466 and Qinglan Project for excellent youth or middle-aged academic leaders in Jiangsu Province (China).

References:

Afzalirad, M., & Shafipour, M. (2018). Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions. *Journal of Intelligent Manufacturing*, 29(2), 423-437. https://doi.org/10.1007/s10845-015-1117-6.

Akyol Ozer, E., & Sarac, T. (2019). MIP models and a matheuristic algorithm for an identical parallel machine scheduling problem under multiple copies of shared resources constraints. *TOP*, 27(1), 94-124. https://doi.org/10.1007/s11750-018-00494-x.

Blazewicz, J. (1981). Solving the resource constrained deadline scheduling problem via reduction to the network flow problem. *European Journal of Operational Research*, 6(1), 75–79. https://doi.org/10.1016/0377-2217(81)90331-3.

Burke, E., Hellier, R., Kendall, G., & Whitwell, G. (2006). A New Bottom-Left-Fill Heuristic Algorithm for the Two - Dimensional Irregular Packing Problem. *Operations Research*, 54(3), 587-601. https://pubsonline.informs.org/doi/abs/10.1287/opre.1060.0293.

Chen, L., Ye, D., & Zhang, G. (2018). Parallel machine scheduling with speed-up resources. *European Journal of Operational Research*, 268(1), 101-112. https://doi.org/10.1016/j.ejor.2018.01.037.

Edis, E.B., Oguz, C., & Ozkarahan, I. (2012). Solution approaches for simultaneous scheduling of jobs and operators on parallel machines. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 27(3), 527–535.

Edis, E.B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research*, 230(3), 449-463. https://doi.org/10.1016/j.ejor.2013.02.042.

Fanjul-Peyro, L., Perea, F., & Ruiz, Rubén. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *European Journal of Operational Research*, 260(2), 482-493. https://doi.org/ 10.1016/j.ejor.2017.01.002.

Fu, Y., Jiang, G., Tian, G., & Wang, Z. (2019). Job scheduling and resource allocation in parallel‐machine system via a hybrid nested partition method. *IEEJ Transactions on Electrical and Electronic Engineering*, 14(4), 597-604. https://doi.org/10.1002/tee.22842.

Gyorgyi, Péter. (2017). A PTAS for a resource scheduling problem with arbitrary number of parallel machines. *Operations Research Letters*, 45(6), 604-609. https://doi.org/10.1016/j.orl.2017.09.007.

Holland, & John, H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2), 88-105. https://doi.org/10.1137/0202009.

Jin, J., & Ji, P. (2017). Scheduling jobs with resource-dependent ready times and processing times depending on their starting times and positions. *The Computer Journal*, 61(9), 1323-1328. https://doi.org/10.1093/comjnl/bxx120.

Lann, A., & Mosheiov, G. (2003). A note on the maximum number of on-time jobs on parallel identical machines. *Computers & Operations Research*, 30(11), 1745–1749. https://doi.org/10.1016/S0305-0548(02)00084-9.

Lee, W. C., Chuang, M. C., & Yeh, W. C. (2012). Uniform parallel-machine scheduling to minimize makespan with position-based learning curves. *Computers & Industrial Engineering*, 63(4), 813–818. https://doi.org/10.1016/j.cie.2012.05.003.

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of Machine Scheduling Problems. *Annals of Discrete Mathematics*, 1(4), 343–362. https://doi.org/10.1016/S0167-5060(08)70743-X.

Liang, X., Zhou, S., Chen, H., & Xu, R. (2019). Pseudo transformation mechanism between resource allocation and bin-packing in batching environments. Future

Generation Computer Systems. 95(JUN), 79-88. https://doi.org/10.1016/j.future.2019.01.006.

McNaughton, R. (1959). Scheduling with Deadlines and Loss Functions. *Management Science*, 6(1), 1–12. http://www.jstor.org/stable/2627472.

Mokotoff, E., & Chrétienne, P. (2002). A cutting plane algorithm for the unrelated parallel machine scheduling problem. *European Journal of Operational Research*, 141(3), 515–525. https://doi.org/10.1016/S0377-2217(01)00270-3.

Peng, B.T., & Zhou, Y.W. (2012). Recursive Heuristic Algorithm for the 2D Rectangular Strip Packing Problem. [In Chinese.] *Journal of Software*. 23(10), 2600–2611. https://doi.org/10.3724/SP.J.1001.2012.04187.

Pinedo, M.L. (2016). Scheduling: Theory, Algorithms and Systems. 5th ed. New York, USA: Springer. https://doi.org/10.1007/978-3-319-26580-3.

Cox, D.R., & Quenouille, M.H. (1953). The Design and Analysis of Experiment. *Biometrika*, 40(3/4), 471-472. http://www.jstor.org/stable/2333370.

Rubén Ruiz, & Concepción Maroto. (2006). A genetic algorithm for hybrid flow shops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3), 781-800. https://doi.org/10.1016/j.ejor.2004.06.038.

Slowinski, R. (1980). Two approaches to problems of resource allocation among project activities – a comparative study. *Journal of the Operational Research Society*, 31(8), 711–723. https://doi.org/10.1057/jors.1980.134.

Ta, Q.C., Billaut, J.-C., & Bouquard, J.-L. (2015). Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem. *Journal of Intelligent Manufacturing*, 29(3), 617-628. https://doi.org/10.1007/s10845-015-1046-4.

Ventura, J.A., & Kim, D. (2000). Parallel machine scheduling about an unrestricted due date and additional resource constraints. *IIE Transactions*, 32(2), 147–153. https://doi.org/10.1023/a:1007662314880.

Wang, Z., Xiao, C., Lin, X., & Lu, Y. (2017). Single Machine Total Absolute Differences Penalties Minimization Scheduling with a Deteriorating and Resource-Dependent Maintenance Activity. *The Computer Journal*, 61(1), 105-110. https://doi.org/10.1093/comjnl/bxx044.